IBM Informix Lab Manual

Informix Quick Start Guide

A no frills step by step guide for installing, initializing and administering Informix based on the things which have to cover with Lab Exercise and Hands-On. The title states Innovator-C specifically, but most everything I talk about here will pertain to all versions of Informix.

In this Lab Manual we will cover:-

- Install 32-bit Informix 12.10.UC7IE Innovator-C Edition on Linux on a server named testsvr01
- Use cooked chunks (filesystem files for data vs. raw disk)
- Initialize an engine named testsvr01 on a server named testsvr01 with 2 GB root dbspace which includes a 1 GB physical log
- Configure 2 GB of logical log space made up of 16 131064 KB logical logs
- Configure System and Logical Log backups using ontape to directories
- Perform a Cold Restore of the last Level 0 backup and applying the Logical Log backups
- Add a 4GB dbspace for data and indexes
- Optionally add a 4GB chunk to existing dbspace

Installing Informix

1. Download Informix Innovator-C from IBM Informix Downloads page

2. Create the informix group and informix user

```
root> groupadd informix
root> useradd -g informix -m informix
root> passwd informix
```

3. Create INFORMIXDIR

```
root> mkdir /opt/informix-ids-12.10.UC7IE
root> chown informix:informix /opt/informix-ids-12.10.UC7IE
root> ln -s /opt/informix-ids-12.10.UC7IE /opt/informix
```

4. Setup environment variables

```
root> vi /etc/profile.d/informix.sh


# location of informix install
export INFORMIXDIR=/opt/informix

# add $INFORMIXDIR/bin to PATH for easy execution of Informix commands
export PATH=${INFORMIXDIR}/bin:${PATH}
```

```
# I still like to set this env variable even tho we used the default location
export INFORMIXSQLHOSTS=${INFORMIXDIR}/etc/sqlhosts

# the name of our Informix instance, matches sqlhosts and DBSERVERNAME in ONCONFIG
export INFORMIXSERVER=blogsvr01

# define the config filename
export ONCONFIG=onconfig.${INFORMIXSERVER}

ESC+:+wq!
```

5. Change to a temporary directory that holds the tarball from step 1 and untar

```
root> cd /root/tmpinfinstall
root> tar -xvf iif.12.10.UC9IE.Linux-RHEL5.tar
```

6. Run the Informix install script ids_install. This will install the engine, the CSDK and JDBC. Answer n when asked to run in GUI mode, accept the License Agreement and accept the defaults for everything else

```
root> . /etc/profile.d/informix.sh
root> ./ids_install
```

Initializing the Engine

Original Blog Post: Initializing Informix Innovator-C on Linux

End Result:

- Informix engine initialized and running
- 2 GB root dbspace
- 1 GB physical log in the root dbspace

Notes:

- Do not use Journaled Filesystems (JFS) for Informix chunks. Read Art Kagel's New Journaled Filesystem Rant

1. Create the chunks directory in a partition that has enough space to hold your data

```
informix> mkdir /home/informix/chunks
```

2. Create the rootdbs chunk named ROOTDBS.01

```
informix> touch /home/informix/chunks/ROOTDBS.01
informix> chmod 660 /home/informix/chunks/ROOTDBS.01
```

3. Create the sqlhosts file

```
informix> vi $INFORMIXSQLHOSTS

# sqlhosts field definitions
# field 1 - DBSERVERNAME
# field 2 - network protocol, onsoctcp is for socket based TCP
# field 3 - hostname or ip address of server running the engine
# field 4 - port number or service from /etc/services
# field 5 - optional options

# blogsvr01
blogsvr01   onsoctcp    blogsvr01   idstcp01
```

4. Create ONCONFIG file and change ROOTPATH, ROOTSIZE, PHYSFILE and DBSERVERNAME

```
informix> cp $INFORMIXDIR/etc/onconfig.std $INFORMIXDIR/etc/$ONCONFIG
informix> vi $INFORMIXDIR/etc/$ONCONFIG

ROOTPATH /home/informix/chunks/ROOTDBS.01
ROOTSIZE 2097152
PHYSFILE 1048576
DBSERVERNAME blogsvr01
```

5. Add idstcp01 port to /etc/services

```
root> vi /etc/serivces

idstcp01        1526/tcp          # Informix
```

6. Add blogsvr01 host in hosts file

root> vi /etc/hosts

ip blogsvr01

6. Initialize the engine

```
informix> oninit -iv
```

Configure Logical Logs

Original Blog Post: Blogging About Logging - Informix Physical and Logical Logging

End Result:

- 2 GB llogdbs01 dbspace
- 16 131064 KB logical logs created in llogdbs01 dbspace
- Remove 6 logical logs created during initialization

1. Temporarily set LTAPEDEV to /dev/null and bounce the engine

```
informix> vi $INFORMIXDIR/etc/$ONCONFIG

LTAPEDEV /dev/null

informix> onmode -k
informix> oninit -v
```

2. Create a new dbspace called lgspace1 to hold the new logical logs

```
informix> touch /home/informix/chunks/LLOGDBS11.01
informix>chown informix:Informix /home/informix/chunks/LLOGDBS11.01
informix> chmod 660 /home/informix/chunks/LLOGDBS11.01
informix> onspaces -c -d lgspace1 -p /home/informix/chunks/LLOGDBS11.01 -o 0 -s 2097152
```

3. Make logical log 4 the current logical log and put the last checkpoint in logical log 4

Execute onmode -l multiple times until the output for logical log 4 in onstat -l shows 'C' in flags

```
informix> onmode -l # repeat as necessary
informix> onmode -c
```

4. Drop logical logs 1, 2 and 3

```
informix> onparams -d -l 1 -y
informix> onparams -d -l 2 -y
informix> onparams -d -l 3 -y
```

5. Create 3 new logical logs

```
informix> onparams -a -d lgspace1 -s 131064
informix> onparams -a -d lgspace1 -s 131064
informix> onparams -a -d lgspace1 -s 131064
```

6. Move the current logical log and last checkpoint out of logical logs 4, 5 and 6

```
informix> onmode -l
informix> onmode -l
informix> onmode -l
informix> onmode -c
```

7. Drop logical logs 4, 5 and 6

```
informix> onparams -d -l 4 -y
informix> onparams -d -l 5 -y
informix> onparams -d -l 6 -y
```

8. Create logical logs 4 through 16

```
informix> onparams -a -d lgspace1 -s 131064 # repeat 12 times
```

Configure System Backups to a Directory

Original Blog Post: **Informix Backup and Restore**

End Result: System Backups written to /home/informix/backup/system

1. Create a directory to hold the System Backups

```
informix> mkdir /home/informix/backup
informix> mkdir /home/informix/backup/system
informix> chmod 770 /home/informix/backup/system
```

2. Change ONCONFIG TAPEDEV parameter to new backup directory

```
informix> vi $INFORMIXDIR/etc/$ONCONFIG

TAPEDEV /home/informix/backup/system
```

3. Optionally compress System Backups

```
informix> vi $INFORMIXDIR/etc/$ONCONFIG

BACKUP_FILTER /bin/gzip
RESTORE_FILTER /bin/gunzip
```

4. Take a Level 0 System Backup

```
informix> ontape -s -L 0
```

Configure Automatic Logical Log Backup to a Directory

Original Blog Post: Informix Backup and Restore

End Result: Automatic Logical Log backups written to /home/informix/backup/llog

1. Create a directory to hold the Logical Log backups

```
informix> mkdir /home/informix/backup/llog
informix> chmod 770 /home/informix/backup/llog
```

2. Change ONCONFIG LTAPEDEV parameter to new backup directory

```
informix> vi $INFORMIXDIR/etc/$ONCONFIG

LTAPEDEV /home/informix/backup/llog
```

3. Optionally change ONCONFIG LTAPESIZE parameter if backup and restore filters are used

```
informix> vi $INFORMIXDIR/etc/$ONCONFIG

LTAPESIZE 2097152
```

4. Modify $INFORMIXDIR/etc/alarmprogram.sh to enable automatic logical log backups with ontape

```
informix> vi $INFORMIXDIR/etc/alarmprogram.sh

# line 31, change flag from N to Y
BACKUPLOGS=Y

# line 62, change onbar -b -l to ontape -a -d
BACKUP_CMD="ontape -a -d"
```

5. Test by forcing a logical log switch

```
informix> onmode -l
```

All used logical logs except for the current logical log should be backed up. Verify by checking for a flag of '**B**' in **onstat -l** output, monitoring the Informix log via **onstat -m** and checking the logical log backup directory for logical log backups.

Restoring from System Backup and Applying the Logical Log backups

Original Blog Post: Informix Backup and Restore

End Result: Informix engine restored to most recent point in time available

1. Verify the engine is offline

```
informix> onstat -
shared memory not initialized for INFORMIXSERVER 'blogsvr01'
```

2. Verify chunks required for restore exist, recreate using touch command if necessary

```
informix> ls -l /home/informix/chunks

total 4194312
-rw-rw---- 1 informix informix 2147483648 2010-07-26 12:35 LLOGDBS01.01
-rw-rw---- 1 informix informix 2147483648 2010-07-26 12:35 ROOTDBS.01

informix> touch /home/informix/chunks/DATADBS01.01 # assuming we need a chunk named DATADBS01.01 and it does
not exist
informix> chmod 660 /home/informix/chunks/DATADBS01.01
informix> ls -l /home/informix/chunks
total 4194312
-rw-rw---- 1 informix informix          0 2010-07-26 12:38 DATADBS01.01
-rw-rw---- 1 informix informix 2147483648 2010-07-26 12:35 LLOGDBS01.01
-rw-rw---- 1 informix informix 2147483648 2010-07-26 12:35 ROOTDBS.01
```

3. Execute a ontape restore from directory

```
informix> ontape -r -d
```

4. Bring the engine into Multi-User mode from Quiescent mode

```
informix> onmode -m
```

Create a Dbspace for Data and Indexes

Original Blog Post: Creating Dbspaces, Databases, Tables and Indexes in Informix

End Result

- 4GB dbspace created for data and indexes

1. Create a new 4GB dbspace

```
informix> touch /home/informix/chunks/DATADBS01.01
informix> chmod 660 /home/informix/chunks/DATADBS01.01
informix> onspaces -c -d datadbs01 -p /home/informix/chunks/DATADBS01.01 -o 0 -s 4194304
```

2. Take a Level 0 backup

```
informix> ontape -s -L 0 -d
```

Optionally Add a Chunk to an Existing Dbspace

Original Blog Post: Creating Dbspaces, Databases, Tables and Indexes in Informix

End Result

- 4GB dbspace added to existing dbspace

1. Add a chunk to an existing dbspace

```
informix> touch /home/informix/chunks/DATADBS01.02
informix> chmod 660 /home/informix/chunks/DATADBS01.02
informix> onspaces -a datadbs01 -p /home/informix/chunks/DATADBS01.02 -o 0 -s 4194304
```

Creating Dbspaces, Databases, Tables and Indexes in Informix

I guess at some point you'd like to actually create a database and create some tables with indexes to hold some data. This logging and backup stuff is fun and all, but what good is a database without data? We know that you put database objects in dbspaces and we have 2 of those already, the rootdbs and llogdbs01, but we don't want to put our data in these dbspaces. If we have multiple disks at our disposal we would put the chunks that make up these dbspaces on different disks to minimize I/O contention and increase parallelism, we would want to do the same thing with our dbspaces that hold our tables and indexes. Even if you are limited to one or two disks it is a good idea to keep your data in separate dbspaces. First of all our llogdbs01 dbspace only has 11 pages available, so we can't fit much data in there and secondly you want to avoid filling up the rootdbs to give Informix space for housekeeping.

Create a new dbspace for data called datadbs01 using the steps we used to create the llogdbs01 dbspace. I'll create this dbspace with 1 4GB chunk, feel free to create a smaller or larger dbspace depending on the amount of storage you need/have available. We will use the default page size of 2K, even though different sizes ranging from 2K to 16K are available, this is a topic for another day.

For performance and data integrity reasons **do not use journaled filesystems** (EXT3, EXT4, ZFS, etc.) for any of your cooked chunks, including the rootdbs and llogdbs01 dbspace chunks we created earlier. Art Kagel explains why in his blog post New Journaled Filesystem Rant.

```
informix> touch /home/informix/chunks/DATADBS01.01
informix> chmod 660 /home/informix/chunks/DATADBS01.01
informix> onspaces -c -d datadbs01 -p /home/informix/chunks/DATADBS01.01 -o 0 -s 4194304
Verifying physical disk space, please wait ...
Space successfully added.

** WARNING **  A level 0 archive of Root DBSpace will need to be done.
```

Take a Level 0 backup like we're asked to and verify the dbspace was added by running onstat -d

```
informix> ontape -s -L 0 -d
informix> onstat -d
IBM Informix Dynamic Server Version 12.10.UC7IE -- On-Line -- Up 01:06:04 -- 144148 Kbytes

Dbspaces
address  number   flags      fchunk   nchunks  pgsize   flags    owner    name
4ae5b808 1        0x40001    1        1        2048     N  B     informix rootdbs
4ae5bc80 2        0x40001    2        1        2048     N  B     informix llogdbs01
4be72410 3        0x40001    3        1        2048     N  B     informix datadbs01
 3 active, 2047 maximum

Chunks
address  chunk/dbs     offset     size       free       bpages     flags pathname
4ae5b968 1      1      0          1048576    518077                PO-B- /home/informix/chunks/ROOTDBS.01
4ae5bde0 2      2      0          1048576    11                    PO-B- /home/informix/chunks/LLOGDBS01.01
4be72570 3      3      0          2097152    2097099               PO-B- /home/informix/chunks/DATADBS01.01
 3 active, 32766 maximum

NOTE: The values in the "size" and "free" columns for DBspace chunks are
      displayed in terms of "pgsize" of the DBspace to which they belong.

Expanded chunk capacity mode: always
```

Easy peasy lemon squeezy.

Now lets create an unbuffered logging database named 'blog' in datadbs01 using dbaccess, a curses based Informix utility for running SQL statements. Typically I would use dbaccess in Menu mode for something like this, but to make the examples clearer I will use dbaccess in Interactive Non-Menu mode (by supplying a hyphen as the second command line argument to dbaccess).

```
informix> dbaccess - -
> create database blog in datadbs01 with log;

Database created.
```

I have successfully created the blog database and am currently connected to it. When I told Informix to create blog in datadbs01 all of the housekeeping stuff for the blog database were put in datadbs01 and datadbs01 will be the default dbspace for any tables, indexes, etc. I create.

Now lets **create some tables**. How about some tables to record blog entries for multiple blogs, call them blog and blog_post. For this type of stuff, things that involve more typing and are a little more complicated, I like to have dbaccess execute a .sql file that I prepare before hand.

```
informix> vi create_tabs.sql

create table blog (
   id              serial not null,
   name            varchar(255)
) in datadbs01 extent size 256 next size 256 lock mode row;

create table blog_post (
   id              serial not null,
   blog_id         integer not null,
   title           varchar(255)
) in datadbs01 extent size 1024 next size 1024 lock mode row;

informix> dbaccess blog create_tabs.sql

Database selected.


Table created.
Table created.


Database closed.
```

If you're familiar with databases, create_tabs.sql should look familiar enough except for the Informix specific stuff. The Serial data type is an auto-incrementing integer, in datadbs01 puts the table in the datadbs01 dbspace, extent size and next size specify the first and next extent sizes in KB and finally lock mode row enables row level locking on this table vs. page level locking.

Looks like we need some referential constraints and some indexes on those tables. You could do this as part of the table creation, but if you do Informix will name the supporting indexes all funky and if you have to do something with them later it will be a pain. Because of this I like to manually create the indexes and create foreign and primary keys afterwards.

```
informix> vi create_idx.sql

create unique index blog_pk on blog (id) in datadbs01;
alter table blog add constraint primary key (id)
    constraint blog_pk;

create unique index blog_post_pk on blog_post (id) in datadbs01;
alter table blog_post add constraint primary key (id)
    constraint blog_post_pk;

create index blog_post_fk1 on blog_post (blog_id) in datadbs01;
alter table blog_post add constraint foreign key (blog_id)
    references blog (id)
    constraint blog_post_fk1;

informix> dbaccess blog create_idx.sql

Database selected.


Index created.


Table altered.


Index created.


Table altered.


Index created.


Table altered.


Database closed.
```

I created 2 unique indexes with 2 primary keys on top of them and 1 non unique index with a foreign key on top of it. This way requires a little bit more typing, but it is how I like to do it and if you want to put the underlying referential constraint indexes in a non default dbspace and give these indexes a name then you should do it this way too.

Time to put some data in these tables. I'm going to use a INSERT INTO ... VALUES SQL to put data into the blog table and use a dbaccess LOAD FROM .. INSERT INTO SQL to load data from a pipe delimited file into blog_post.

```
informix> vi load.sql
```

```
insert into blog (id, name) values (0, "Informix DBA");
insert into blog (id, name) values (0, "Informix technology");

select * from blog;

informix> dbaccess blog load.sql

Database selected.


1 row(s) inserted.


1 row(s) inserted.




id    1
name  Informix DBA

id    2
name  Informix technology

2 row(s) retrieved.


Database closed.

informix> vi load.unl

0|1|Creating Dbspaces, Databases, Tables and Indexes in Informix|
0|1|ZOMG, FYI - IM Informix Tech Support FTW|
0|1|Informix Backup and Restore - The Bare Minimum|
0|2|A bug can undermine your troubleshooting|
0|2|Informix Editions revisited|
0|2|New Informix editions: Bargain time?|

informix> dbaccess blog -

Database selected.

> load from load.unl insert into blog_post;

6 row(s) loaded.

> select * from blog_post;

id       1
blog_id  1
title    Creating Dbspaces, Databases, Tables and Indexes in Informix

id       2
blog_id  1
title    ZOMG, FYI - IM Informix Tech Support FTW

id       3
blog_id  1
title    Informix Backup and Restore - The Bare Minimum

id       4
blog_id  2
title    A bug can undermine your troubleshooting

id       5
```

```
blog_id  2
title    Informix Editions revisited

id       6
blog_id  2
title    New Informix editions: Bargain time?

6 row(s) retrieved.
```

The last thing I want to do is show you how to add a chunk to an existing dbspace, something you might want to do if you need more space for your data. This is very similar to creating a dbspace and is done via the same onspaces command

```
informix> touch /home/informix/chunks/DATADBS01.02
informix> chmod 660 /home/informix/chunks/DATADBS01.02
informix> onspaces -a datadbs01 -p /home/informix/chunks/DATADBS01.02 -o 0 -s 4194304
Verifying physical disk space, please wait ...
Chunk successfully added.

informix> onstat -d

IBM Informix Dynamic Server Version 12.10.UC7IE -- On-Line -- Up 00:26:37 -- 144148 Kbytes

Dbspaces
address  number   flags      fchunk   nchunks  pgsize   flags    owner    name
4ae5b808 1        0x60001    1        1        2048     N  B     informix rootdbs
4ae5bb88 2        0x40001    2        1        2048     N  B     informix llogdbs01
4bd65d28 3        0x60001    3        2        2048     N  B     informix datadbs01
 3 active, 2047 maximum

Chunks
address  chunk/dbs     offset     size       free       bpages     flags pathname
4ae5b968 1      1      0          1048576    518259                PO-B- /home/informix/chunks/ROOTDBS.01
4ae5bce8 2      2      0          1048576    11                    PO-B- /home/informix/chunks/LLOGDBS01.01
4bb51cc8 3      3      0          2097152    2095214               PO-B- /home/informix/chunks/DATADBS01.01
4bdb73e0 4      3      0          2097152    2097149               PO-B- /home/informix/chunks/DATADBS01.02
 4 active, 32766 maximum

NOTE: The values in the "size" and "free" columns for DBspace chunks are
      displayed in terms of "pgsize" of the DBspace to which they belong.

Expanded chunk capacity mode: always
```

Informix Backup and Restore - The Bare Minimum

Christian Szell: Is it safe?... Is it safe?

Babe: You're talking to me?

Christian Szell: Is it safe?

Babe: Is what safe?

Christian Szell: Is it safe?

Babe: I don't know what you mean. I can't tell you something's safe or not, unless I know specifically what you're talking about.

Christian Szell: Is it safe?

Babe: Tell me what the "it" refers to.

Christian Szell: Is it safe?

Babe: Yes, it's safe, it's very safe, it's so safe you wouldn't believe it.

Christian Szell: Is it safe?

Babe: No. It's not safe, it's... very dangerous, be careful.

- Marathon Man, 1976

Hopefully you will never need to restore from a backup, but chances are you will and it can be as scary as dental torture. Disks fail, servers fail, disasters destroy data centers, users accidentally delete or update the wrong data and other generally sucky things happen to the databases we administer. Day in and day out database backups are annoying little things that suck up I/O resources and storage space only to be discarded when the next backup comes along, but there is no better feeling than being able to restore the last backup and get your system back online with minimal data loss after a failure. There is no worse feeling than needing a backup and not having one. The good news is basic Informix backup and restore is pretty simple and straight forward in recent versions of Informix.

I'm going to assume you don't have or don't want to use a tape storage device and a Storage Manager. If this does not describe you, then this isn't the post for you. You'll want to take a look at onbar and a Storage Manager like ISM or TSM. I'm assuming you just want to backup your data to the filesystem and you'll do the due diligence of moving these backup files off site or at least to a different server/disks to provide maximum recoverability. For this we will use the ontape Informix utility and the (relatively) new feature of automatically backing up and restoring from a directory.

Some of what follows comes from a great blog by Fernando Nunes at Informix Technology that I used as a resource for understanding ontape backup and restore to directories.

There are 3 kinds of backups, System Backups, Logical Log Backups and Critical File Backups.

System Backups

A System Backup is a direct copy of the Informix pages that make up your engine. This is done with the engine online and with no impact or blocking of users other than the additional I/O strain put on the system to read each page from the engine and write out each page to disk.

There are 3 levels of System Backup that can be taken:

- A Level 0 backs up every page
- A Level 1 backs up every page that has changed since the last Level 0 backup
- A Level 2 backs up every page that has changed since the last Level 1 backup

If you use multiple backup levels, when it comes time to restore you would first restore the last Level 0 then apply the most recent Level 1 backup (if one exists) after the Level 0 and finally apply the most recent Level 2 backup (if one exists) after the Level 1 backup is restored.

Using Level 1 and Level 2 backups can reduce the size of your backups taken and the I/O resources required to write backups to disk (less data to write, we still have to read all of the data), but we're here for the bare minimum so I'm just going to use Level 0 backups and Logical Log backups to keep our data safe.

Using a System Backup to restore data is called a Physical Restore.

Logical Log Backups

If you are not familiar with Logical Logging check out a prior post on logging.

Each logical log should be backed up when it becomes full. Not only because it will help you recover lost data but because Informix will BLOCK if it needs to reuse a logical log that has not been backed up (remember the logical logs are used in a cyclical fashion) and it will continue to block until the needed logical log is backed up. Informix does this because it assumes data recoverability is more important than engine availability. Whether you agree with this philosophy or not it is the way the engine works, so you will need to backup your logical logs to prevent the engine from blocking.

There is a work around, but it is not recommended. You can set LTAPEDEV to /dev/null in the ONCONFIG file which will disable logical log backups and automatically mark a logical log as backed up when it stops being the current logical log. If you do this your recovery from a failure will be limited to the data stored in your Level 0, 1 and 2 System Backups.

The logical log backups are used during a restore to replay transactions that have occurred since the last System Backup. After the Physical Restore of a Level 0 (and optionally a Level 1 and 2) backup you perform a Logical Restore of the backed up logical logs.

Logical Log Salvage

If your engine crashes and needs to be restored there will be at least one logical log that has not been backed up yet, the current logical log. It would be real nice to be able to backup this logical log before we try and restore so it can be in the list of logical log backups to restore our engine to the most up to date version before the crash. Well the engineers that implemented Informix were real smart and gave us this ability. Before a Physical Restore is attempted you will be asked if you want to backup the logical logs and salvage them before we overwrite them with a restore. This works great as long as the disk that holds the logical logs isn't the thing causing you to restore. See the logging post about sizing your logical logs appropriately to minimize data loss in this scenario.

Critical File Backups

There are some files that you should backup each time you take a System Backup, these files will help you recreate your Informix install in the event that the filesystem that holds your Informix install and config files becomes unavailable

- $INFORMIXDIR/etc/$ONCONFIG
- $INFORMIXSQLHOSTS or $INFORMIXDIR/etc/sqlhosts
- $INFORMIXDIR/etc/oncfg_*
- onstat -d output, to give you a listing of the chunk paths if you need to recreate them

Unfortunately there is no Informix utility for this, you should write a script to tar.gz these files and move them somewhere safe each time a System Backup is taken. Something like this should get you pointed in the right direction.

```
informix> onstat -d > /home/informix/tmp/onstat.d.out
informix> tar -czf /home/informix/backup/files/criticalfiles.$(date "+%Y%m%d.%H%M").tar.gz
$INFORMIXDIR/etc/$ONCONFIG $INFORMIXSQLHOSTS $INFORMIXDIR/etc/oncfg_* /home/informix/tmp/onstat.d.out
```

Taking a System Backup

We will be using the Informix utility ontape and its ability to backup to a directory for all of our System and Logical Log backup needs.

To enable ontape System Backups to a directory, modify the ONCONFIG parameter TAPEDEV to a directory name that is owned by user informix and group informix and has read, write, execute permissions for user and group. This is where the System Backups will live.

```
informix> mkdir /home/informix/backup
informix> mkdir /home/informix/backup/system
informix> chmod 770 /home/informix/backup/system
informix> vi $INFORMIXDIR/etc/$ONCONFIG
```

```
TAPEDEV /home/informix/backup/system
```

The most recent System Backup will have a name of DBSERVERNAME_SERVERNUM_L[0, 1 or 2]. A Level 0 backup for our blogsvr01 engine will be /home/informix/backup/system/blogsvr01_0_L0

Each time you execute a new System Backup the old backup is renamed with a timestamp inserted between SERVERNUM and L[0, 1 or 2]

A listing of all the backups of my blogsvr01 engine looks like this

```
blogsvr01_0_20100723_133319_L0
blogsvr01_0_20100725_135325_L0
blogsvr01_0_L0
```

To actually take a Level 0 backup

```
informix> ontape -s -L 0 -d
```

-s tells ontape we want a System Backup, -L 0 asks for a Level 0 (-L 1 and -L 2 for Level 1 or 2 backups) and -d says we're backing up to a directory and prevents ontape from prompting us for any input.

I recommend executing this in a script that also backs up the critical files via cron or some other scheduling mechanism, simply remembering to take a backup every day or week and kicking it off manually isn't good enough.

Taking Logical Log Backups

There are multiple ways to use ontape to backup logical logs to a directory, but I think the best way is to use the ALARMPROGRAM, a script called by Informix when events occur, to back them up as they fill.

Your ONCONFIG ALARMPROGRAM parameter should already be set to $INFORMIXDIR/etc/alarmprogram.sh, all we have to do slightly modify this IBM supplied script and set LTAPEDEV in the ONCONFIG to a directory.

```
informix> mkdir /home/informix/backup/llog
informix> chmod 770 /home/informix/backup/llog
informix> vi $INFORMIXDIR/etc/$ONCONFIG

LTAPEDEV /home/informix/backup/llog

informix> vi $INFORMIXDIR/etc/alarmprogram.sh

# line 31, change flag from N to Y
BACKUPLOGS=Y

# line 62, change onbar -b -l to ontape -a -d
BACKUP_CMD="ontape -a -d"
```

That's is, you shouldn't have to do anything else. Each time a logical log fills up alarmprogram.sh will be called and any unbacked up logical logs will be backed up to your backup/llog directory.

Restoring an Engine

There are many flavors of restore and you should get familiar with them all in the Backup and Restore guide contained in the free online documentation. I will focus on a Cold Restore, which requires the engine to be offline. Presumably because something terrible happened and you need to restore to recover from a server or disk problem.

Using the output from the onstat -d you saved during your last system backup, verify that the files for your chunks exist. If you replaced a failed drive or are restoring to a different server they may not exist, in this case you can use the touch command to recreate a 0 byte file placeholder with the appropriate ownership and permissions. The restore will do the rest.

Using the ontape utility, invoke the restore command with the -d directory option. When -d is used you are not prompted for any decisions, ontape figures out what to do based on what is in the backup directories specified in TAPEDEV and LTAPEDEV.

```
informix> ontape -r
Restore is using file /home/informix/backup/data/blogsvr01_0_L0 ...
```

```
Archive Tape Information

Tape type:      Archive Backup Tape
Online version: IBM Informix Dynamic Server Version 12.10.UC7IE
Archive date:   Sun Jul 25 16:24:17 2010
User id:        informix
Terminal id:    /dev/pts/0
Archive level:  0
Tape device:    /home/informix/backup/data/
Tape blocksize (in k): 32
Tape size (in k): system defined for directory
Tape number in series: 1

Spaces to restore:1 [rootdbs                                                            ]
2 [llogdbs01                                                           ]

Archive Information

IBM Informix Dynamic Server Copyright 2001, 2010  IBM Corporation.
Initialization Time       07/23/2010 15:50:16
System Page Size          2048
Version                   18
Index Page Logging        OFF
Archive CheckPoint Time   07/25/2010 16:13:05

Dbspaces
number   flags    fchunk   nchunks  flags    owner                 name
1        40001    1        1        N  B     informix              rootdbs

2        40001    2        1        N  B     informix              llogdbs01



Chunks
chk/dbs offset   size     free     bpages   flags pathname
1   1   0        1048576  518117            PO-B  /home/informix/chunks/ROOTDBS.01
2   2   0        1048576  425969            PO-B  /home/informix/chunks/LLOGDBS01.01

Continue restore? (y/n)y
Do you want to back up the logs? (y/n)y
File created: /home/informix/backup/llog/blogsvr01_0_Log0000000040
Log salvage is complete, continuing restore of archive.
Restore a level 1 archive (y/n) y
Ready for level 1 tape
Restore is using file /home/informix/backup/data/blogsvr01_0_L1 ...
File /home/informix/backup/data/blogsvr01_0_L1 not found, continuing ...
Do you want to restore log tapes? (y/n)y

Roll forward should start with log number 39
Restore is using file /home/informix/backup/llog/blogsvr01_0_Log0000000039 ...
Rollforward log file /home/informix/backup/llog/blogsvr01_0_Log0000000039 ...
Rollforward log file /home/informix/backup/llog/blogsvr01_0_Log0000000040 ...

Program over.
```

After ontape -r -d completes the engine will be in Quiescent (Single User) mode, resolve this by executing onmode -m to switch to Multi-User Mode and finalize the restore.

```
informix> onmode -m
```

Keeping the Size of Backups In Check

If you would like to reduce the amount of space these backups are taking you're probably thinking about compressing them with gzip or something similar and decompressing the required files before a restore. Well those Informix guys and gals did it again by adding 2 ONCONFIG parameters that allow you to automatically compress and decompress these backup files on the fly via backup and restore filters.

If I would like to use gzip to compress my backups and gunzip to decompress I would modify the BACKUP_FILTER and RESTORE_FILTER ONCONFIG parameters. I will also be required to change LTAPESIZE from the default of 0 (unlimited) to something not 0. Not sure exactly why this is required, but the engine complains if LTAPESIZE is 0.

```
informix> vi $INFORMIXDIR/etc/$ONCONFIG
```

```
LTAPESIZE 2097152
BACKUP_FILTER /bin/gzip
RESTORE_FILTER /bin/gunzip
```

Increasing Buffers and Reducing I/O with Informix

If you're following along with my posts on installing, initializing, adding space and creating databases and tables in Informix then you're ready to start doing some basic performance tuning. You can improve performance dramatically by making some simple changes to the ONCONFIG and bouncing the engine. The first and most important change to make is to increase Buffers, this is where Informix will be caching all of the data you are reading in from disk. More data cached means less I/O and less I/O has to mean better performance and since I/O will most likely be your current bottleneck, reducing I/O should result in the most performance gain.

How Informix Caches Data

When you execute SQL Informix determines what pages on disk it needs to read or write to satisfy this SQL request. Informix will first check to see if a page it needs is in memory and if it is not it will read the page from disk and put it into memory. This memory cache is made up of many buffers called a buffer pool. Each buffer is the size of a page on disk. So far we've only used the default page size of 2 KB for our dbspaces, so our buffer pool is made up of many 2 KB buffers.

LRU Queues

The size of the buffer pool is static and created at engine start up. After some time the buffer pool will become full of pages read in from disk and new reads will need to kick some pages out of memory to make room. Informix doesn't just arbitrarily pick pages at random to kick out of the pool, that would be stupid. Informix assumes that the pages in memory that have not been used in a while aren't likely to be used again anytime soon and chooses these pages to be replaced by the new pages you need in the buffer pool.

The bufffers in the buffer pool are divided evenly among multiple Least Recently Used Queues (LRU), when a buffer is used by a SQL request it is moved to the end of the queue putting the most recently used pages at the tail of the queue. When a page needs to to be kicked out of the buffers a page at the head of a LRU queue is picked. For the record, I am oversimplifying the LRU queues, there is more going on here like FIFOs, buffer priority and Free/Modified LRU queues that make up a LRU queue, but for now this is all you need to know about the anatomy of a LRU queue. Read more in the free Online Documentation.

Dirty Pages in LRU Queues

When Informix needs to kick a page out of memory to make room for a new page it prefers to pick a page that has not been modified. When Informix does this it can simply replace the data in the buffer with the new data. Since this page has not been modified it doesn't need to do anything else, if the page had been modified Informix would need to write the changes to disk. When there are no unmodified pages in the LRU queue Informix performs a Foreground Write, a single write of a modified page to disk, before it replaces the data in the buffer. These are bad for performance and you don't want to see them. Monitor Foreground Writes by executing onstat -F, the number of Foreground Writes will appear at the head of this output.

```
informix> onstat -F
IBM Informix Dynamic Server Version 12.10.UC7IE -- On-Line -- Up 2 days 18:22:18 -- 144148 Kbytes


Fg Writes     LRU Writes    Chunk Writes
7             146           3169

address  flusher  state    data     # LRU    Chunk    Wakeups  Idle Tim
4aefb620 0        I        0        0        347      239020   238928.516
4aefbc28 1        I        0        0        1        238674   238927.649
```

Avoiding Foreground Writes

You can prevent Foreground Writes (FG) by configuring LRU writes. A LRU write is more efficient than a FG write because it is performed by a page cleaner thread in the background and not by your session thread as in a FG write. LRU writes are triggered when a LRU queue reaches a maximum percent dirty that you configure and continue writing modified pages to disk until the LRU queue reaches a minimum percent dirty that you configure.

LRU writes also minimizes the number of pages written to disk at Checkpoint time. During a checkpoint all modified pages are written to disk to provide a consistent point of recovery in the event of a failure. Keeping the number of pages written during a checkpoint low is not as important as it used to be before non blocking checkpoints came around. During a blocking checkpoint no pages can be modified so all sessions attempting to insert, update or delete data block until the checkpoint completes. This behavior resulted in many OLTP engines getting configured with LRU max dirty of less than 1% to keep the checkpoint times short. With the introduction of non blocking checkpoints checkpoint time isn't as much of a concern and LRU max dirty of 20% to 70% is more common.

How Many Buffers Do I Need?

The more the merrier, until you reach the point of dimishining returns. If your data size is only 1 GB then a buffer pool of 2 GB is just going to be 1 GB of wasted memory. Also, if you only routinely access 1 GB of a 100 GB data set then 1 GB of your 2 GB buffer pool will be holding cached data that isn't accessed very often. Figuring out the optimal buffer pool size for your data and your available memory resources is a balancing act.

If you are using the free for production release of Informix called Innovator-C you are limited to 2 GB of shared memory for the engine. This shared memory limitation applies to all of the memory Informix uses, not just buffers. Typically the memory Informix uses is made up of 2 areas, the Resident portion which includes the buffer pool and the Virtual portion which includes memory required for session threads, sorting, grouping and other things. An optional 3rd and much smaller portion is created if you use shared memory connections (we have not configured any of these yet). For starters we will configure 50% of our available 2 GB for buffers. At 2 KB per buffer this is 524288 buffers.

How Many LRU Queues Do I Need?

There are many schools of thought on this one. The manuals say at least 4 LRU queues and no more than CPUVPs configured. For Innovator-C that is limited to a max of 4 CPUVPs this is a recommended setting of 4. The default ONCONFIG setting contradicts the manuals and suggests a value of 8 for the 1 CPUVP it has configured. I think these are too low because the penalty for under configuring LRUs is greater than the penalty for over configuring LRUs when you have a lot of concurrent sessions. An older suggestion is to configure 2000 buffers per LRU, but I think this results in too many LRUs. The real answer is to pick something in the middle and monitor the Bufwaits Ratio (BR), a formula devised by the infamous Art Kagel that can be calculated from the onstat -p output.

```
informix> onstat -p
IBM Informix Dynamic Server Version 12.10.UC7IE -- On-Line -- Up 2 days 20:00:34 -- 144148 Kbytes

Profile
dskreads   pagreads   bufreads   %cached dskwrits   pagwrits   bufwrits   %cached
1220       1295       219268     99.45   5449       9476       44802      87.84

isamtot    open       start      read       write      rewrite    delete     commit     rollbk
143006     11032      14820      56477      12433      845        337        2643       3
```

```
gp_read    gp_write   gp_rewrt   gp_del     gp_alloc   gp_free    gp_curs
0          0          0          0          0          0          0

ovlock     ovuserthread ovbuff     usercpu  syscpu   numckpts   flushes
0          0             0         6.03     32.56    704        356

bufwaits   lokwaits   lockreqs   deadlks    dltouts    ckpwaits   compress   seqscans
44         0          62182      0          0          6          1449       3724

ixda-RA    idx-RA     da-RA      RA-pgsused lchwaits
51         0          13         64         11123
```

Bufwaits Ratio (BR) = (bufwaits / (pagreads + bufwrits)) * 100 or (44 / (1295 + 44802)) * 100 = 0.095%

Shoot for a BR less than 7% (lower the better) adding LRUs as needed. We'll start with 63 LRUs.

How Dirty Should My LRU Queues Get?

Like I said earlier, with the addition of non blocking checkpoints super dirty LRU queues that are flushed at checkpoint time are not as big of a deal as they used to be. Checkpoint writes are more efficient than LRU writes and LRU writes are more efficient than Foreground writes, so the goal is to encourage Checkpoint writes while limiting Foreground writes via LRU writes in between checkpoints. There is no formula for figuring this out, you will have to monitor how buffers are being flushed to disk via onstat -F and adjust LRU min/max dirty accordingly. Flushing more aggressively when we have Foreground writes and less agressively when we don't. We'll start with a LRU max dirty of 60% and a LRU min dirty of 50% (which happen to be the default values).

Making the Configuration Changes

Now for the easy part. These configuration changes are made by modifying the BUFFERPOOL ONCONFIG parameter and bouncing the engine.

```
informix> vi $INFORMIXDIR/etc/$ONCONFIG

BUFFERPOOL      size=2K,buffers=524288,lrus=63,lru_min_dirty=50.000000,lru_max_dirty=60.000000

informix> onmode -ky
informix> oninit -v
```

Do I have the Right Number of Buffers Configured?

Scenario 1: Too many buffers configured

If you have too many buffers then you are needlessly wasting memory that could be used for other things (Informix sorting, grouping and session memory or the OS and other applications).

The first place to check is the onstat -P output after your engine has been running under normal load for a while. If you have more buffers configured than you have data then there will be some buffers that are marked as Other, these are the unused buffers.

```
informix> onstat -P | tail -5
Percentages:
Data  32.04
Btree 18.02
Other 49.94
```

In this example 49.94% of my buffer pool is unused, I can probably safely reduce the size of my buffer pool by 50% and not suffer any loss in performance.

If all of your buffers are in use and you need to reduce the memory footprint of your Informix engine you can use the output of onstat -p to monitor your Read Cache Hit % (this will be the first output labeled %cached and is the number of times you needed to read a page and it was in memory, the second %cached is your Write Cache Hit %). If this number is very high (above 98%) you might be able to reduce the number of buffers without suffering much performance loss. However, I would only reduce buffers if I couldn't reduce shared memory any where else.

Scenario 2: Not enough buffers configured

This is the most likely scenario as your working set of data will typically be larger than your buffer pool. Use the same onstat

-p command to monitor the Read Cache Hit % and continue to add buffers until you reach an acceptable percentage. 95% or better is great, above 90% is good and anything less than 90% is probably a sign of needing more buffers.

That's it for basic buffer tuning, not much to it really.

IBM Informix Logging Types & Their Importance

What rolls down stairs alone or in pairs
Rolls over your neighbor's dog?

What's great for a snack and fits on your back?
It's Log, Log, Log!

It's Log, Log. It's big, it's heavy, it's wood.
It's Log, Log. It's better than bad, it's good!

Everyone wants a Log! You're gonna love it, Log!
Come on and get your Log! Everyone needs a Log!

- Ren and Stimpy

Logical Logging

Each time you insert, update or delete a row a logical log record containing information about what was changed and how is written to the logical log (if transaction logging is enabled for your database. You do have this enabled, right?) Each time you create or alter a database object (table, index, sequence, etc.) a logical log record is written to the logical log (even if transaction logging is disabled). Each time you run a backup, a checkpoint happens, you add a chunk or dbspace or a new extent is allocated a logical log record is written to the logical log.

So what are all of these logical log records good for? Well, they're good for a lot of things.

- Transaction Rollback
- Recovery from Engine Failure (Data Restore and Fast Recovery)
- Deferred Constraint Checking
- Cascading Deletes
- Distributed Transactions (Two Phase Commit)
- Enterprise Replication
- High Availability Data Replication

Physical Logging

The Physical Log stores a before-image of any page that Informix is going to modify. A before-image of a page is how the page looked at the time of the last checkpoint. Only one before-image for each modified page is written to the Physical Log. If you were to modify the same page over and over and over again in between checkpoints, only the first modification would cause a before-image to be written to the Physical Log.

The before-images in the Physical Log are used by fast recovery and database restores to give an accurate snapshot of your data at the time of the last checkpoint.

Thatsalotta I/O

Yes, it is and if not configured correctly logical and physical logging can become a bottleneck. Heck, even if configured the right way logical and physical logging can be your bottleneck. Don't be sad, something has to be a bottleneck and it will most likely be the slowest part of a database system, the disk. The goal is to minimize the impact of all this logging. Whether you like it or not, we need to do all of this logging to keep our data safe.

LOGBUFF and PHYSBUFF

When the engine writes to the logical or physical log it doesn't write directly to disk, it puts as many of these writes as it can into one of the 3 logical log buffers or one of the 2 physical log buffers. These buffers are then written to stable storage on disk in a block which is faster than doing them one at a time.

I have never personally seen a performance problem that was a result of poorly configured log buffers nor have I seen much performance gain from dramatically increasing the size of my log buffers. Start out with a LOGBUFF of 64 KB and a PHYSBUFF of 128 KB (the defaults in onconfig.std) and monitor the pages/io fields in the onstat -l output and shoot for values that equal buffer size / 2 * .75 or 75% full at the time of buffer flushing.

NOTE: If Unbuffered logging is configured (this is what I use for the highest level of data integrity) you will be hard pressed to get near 75% buffer fullness at the time of flush if you are an OLTP environment and have a lot of small transactions. Unbuffered logging forces the logical log buffer to be flushed at the end of a transaction before the transaction can complete. I'm currently getting 1.2 pages/io (2.4 KB) for my logical log buffers.

Get Yourself Some More Disks

Look at what can happen when you modify a row

- Find the page to be modified either through and index or a sequential scan
- Read the data page containing the row to be modified from disk into memory
- Write a before-image to the physical log buffer
- Write a logical log record of the update to the logical log buffer
- Modify the data page in memory
- Flush the physical log buffer to disk
- Flush the logical log buffer to disk
- Flush the data page to disk

That's a lot of disk access to different parts of the disk trying to happen at about the same time. You can really improve performance by putting your physical log, logical logs and data on separate spindles. Is this a requirement to get good performance with Informix? Hell no. All RDBMSs use logging in a similar way and I'll bet that there are plenty of single disk MySQL, SQL Server, DB2, Oracle and Informix production installs out there doing just fine. But if maximum performance is your goal and you can get multiple disks to spread and parallelize your I/O then do it.

Increasing the size of the Physical Log

Until recently it was kind of taboo to have your physical log live in the rootdbs. The mantra was "put rootdbs, the physical log and the logical logs on separate disks" and if you didn't your peers would laugh at you and call you names. Truth is, if you are doing what you're supposed to and do not have any user data living in the rootdbs there really isn't too much I/O to the rootdbs so having the physical log live there doesn't cause much I/O conflict.

On engine initialization the physical log is put into the rootdbs and that is where I like to keep it (provided I move the logical logs somewhere else.) You should, however, increase the size of the physical log. There is absolutely no performance penalty for having a big physical log, none. There are drawbacks to having an undersized physical log.

- Checkpoints firing too frequently due to small physical log hitting 75% full mark often
- Checkpoint blocking due to physical log filling during a checkpoint
- Fast Recovery can take longer if physical log is not large enough

So make it bigger than the default of 50,000 KB. Who cares, disk is cheap. I use a physical log size of 4 GB on my systems and the manual recommends a physical log size of 110% the size of your buffer pool. Pick a size based on the available disk you have and the workload for your environment, there really isn't a one size fits all answer here. Start with 1 GB and go from there if you don't know where to start. Use onparams to change the physical log.

```
informix> onparams -p -s 1048576
```

Moving and Increasing the Size of the Logical Logs

Unlike the physical log, there are multiple logical logs. When the current logical log fills up the engine starts using the next logical log and kicks off a backup for the recently filled logical log. The logical log backup is initiated through the script defined by ALARMPROGRAM in your ONCONFIG or the engine marks it backed up if LTAPEDEV is set to /dev/null. This is very important stuff if you're going to be using Informix, but this post is getting long enough so I'll have to save it for another day.

If you don't change anything in your ONCONFIG when you initialize the engine you get 6 logical log files of 10,000 KB each put in your rootdbs. This simply not enough logical logs, the logical logs are too small and they're living with our physical log so we need to change this.

The amount of logical log space you have is a factor of the number of logical logs you have and the size of each log (obvious?). If you do not have enough logical log space you will run into problems. The logical logs are used in a circular fashion, when a new logical log is needed, Informix starts overwriting the data in the logical log that contains the oldest data as long as it is 1) marked as backed up and 2) does not contain any open transactions. If either 1 or 2 are met, the engine will block until neither 1 or 2 are true. Having sufficient logical log space won't help with 1, but it will help with 2.

As with the physical log, there is no penalty for having too much logical log space, but you need to size your individual logical logs to be big enough that you're not switching logs every 3 seconds but not so big that they never fill up and therefore are never backed up. Having super huge logical logs that contain 3 days worth of transactions before they are backed up exposes you a little bit to more data loss. If your system totally dies and the disk that holds the logical logs is destroyed and you can't perform a log salvage during a cold restore to get the logical log data from the logs that are not backed up, well you just lost 3 days of transactions. TMI?

In my environment I have 31 261628 KB logical logs for a total logical log space of about 8 GB. My logical logs switch about every hour during peak times. Your logical log setup will vary, but for shits and grins lets say we want 2 GB of logical log space divided between 16 logical logs.

We're going to need to create a new dbspace to hold these logical logs. I want it to be exactly 2 GB (2 GB * 1048576 KB = 2097152 KB). We do this with the onspaces command.

```
informix> touch /home/informix/chunks/LLOGDBS01.01
informix> chmod 660 /home/informix/chunks/LLOGDBS01.01
informix> onspaces -c -d llogdbs01 -p /home/informix/chunks/LLOGDBS01.01 -o 0 -s 2097152
```

If we look at the output from onstat -d we can see that we have a dbspace with a single chunks containing 1048576 pages and 1048523 pages free because 53 pages have been reserved by Informix.

1048523 pages / 16 logical logs = 65532 pages per logical logs.

Like I said before, when you initialize the engine you get 6 logical logs numbered 1 to 6. These have to go. They're of a funky size and in a different dbspace and I just can't live with that. I just can't drop them all at once either, because Informix requires me to have at least 3 logical logs at all times. I also can't drop the current logical log or a logical log that has the last checkpoint. Picky, picky.

I will use the onmode -l to force logical log switches until I'm in logical log 4 (look for the C flag of the onstat -l output) and then I'll force a checkpoint with onmode -c. I also temporarily set LTAPEDEV to /dev/null and bounced the engine, letting me fake some required logical log backups. I'm assuming you're working with a new engine and don't care about losing data at this point.

I will use the onparams command to drop the first 3 (leaving me with 3 total) then create 3 logical logs of the new size and in the new dbspace to take their place. I can then drop the remaining 3 funky logical logs and create the remaining 13 logical logs.

```
informix> onparams -d -l 1 -y
informix> onparams -d -l 2 -y
informix> onparams -d -l 3 -y
informix> onparams -a -d llogdbs01 -s 131064 # 65532 pages = 131064 KB
informix> onparams -a -d llogdbs01 -s 131064
informix> onparams -a -d llogdbs01 -s 131064
informix> onmode -l
informix> onmode -l
informix> onmode -l
informix> onmode -c
informix> onparams -d -l 4 -y
informix> onparams -d -l 5 -y
informix> onparams -d -l 6 -y
informix> onparams -a -d llogdbs01 -s 131064
informix> onparams -a -d llogdbs01 -s 131064
informix> onparams -a -d llogdbs01 -s 131064
informix> onparams -a -d llogdbs01 -s 131064
informix> onparams -a -d llogdbs01 -s 131064
informix> onparams -a -d llogdbs01 -s 131064
informix> onparams -a -d llogdbs01 -s 131064
informix> onparams -a -d llogdbs01 -s 131064
informix> onparams -a -d llogdbs01 -s 131064
informix> onparams -a -d llogdbs01 -s 131064
informix> onparams -a -d llogdbs01 -s 131064
informix> onparams -a -d llogdbs01 -s 131064
informix> onparams -a -d llogdbs01 -s 131064
```

Checking onstat -d shows me that I have used all but 11 pages (22 KB) in llogdbs01 and onstat -l shows me my 16 new logical logs.

Take it One Step Further

If you are spreading out the I/O to multiple disks and you have disks to spare, try creating two logical log dbspaces on two different drives and alternate the creation of logical logs between these two dbspaces. For example, create logical log 1 in llogdbs01, logical log 2 in llogdbs02, logical log 3 in llogdbs01, etc. When a logical log switch occurs a backup reads all of the data from the old log while you're busy trying to write transactions to the new log. If these two logical logs are on the same disk then you can see some I/O contention during the backup. Is this required? Nah, just something to keep in the back of your head. There is a good chance in this day and age that you won't even be dealing with individual disks anyway. Instead you're probably just be given a slice of disk on the SAN and all this I/O separation stuff is meaningless. Kind of sad really.

Informix HDR Will Save Your Butt

If I had to guess, I would say that most production database engines utilize RAID technology to protect against the inevitable disk failure and the ones that don't probably should. Disk is cheap and the revenue saved by avoiding an extended outage can be enough to pay for disk mirroring many times over.

If I had to guess again, I would say that not nearly enough production database engines utilize High Availability Data Replication (HDR) to protect against the inevitable server failure. Why is this? Servers can fail too. Sure, servers are more expensive than disks and sure the MTTF is longer than disks but the money lost during an extended outage that could have been avoided with HDR is probably going to be more than the cost of implementing an HDR solution.

HDR continuously replicates the changes made to a Primary server to a Secondary server that can be quickly converted to a Primary if the original Primary fails. As an added bonus, the Secondary server can be used for reads and writes allowing you to make use of this hardware to improve performance instead of letting it sit there idle. You could also implement multiple Remote Standalone Secondary (RSS) or Shared Disk Secondary (SDS) servers to create a grid if your Informix Edition supports this. I'm going to focus on a single HDR Secondary which is available for no cost in Innovator-C.

As with most Informix features, HDR is incredibly easy to configure and does not require much administration.

Get Yourself Some More Hardware

To enable HDR you will need another server. This server should be identical to the Primary server. The Secondary server doesn't have to be identical in every way, but if you expect it to take over during a failure you're going to want the same amount of memory, CPUs, etc. to ensure it can handle the load. Here is what is required of servers participating in HDR: Both servers must run the same Informix version

- Both servers must be able to run the same Informix executable. Ubuntu and Red Hat run the same Informix executable; HP/UX and Red Hat do not. Why would you ever want to do this anyway?
- Both servers must have network capabilities
- The Secondary server must have at least as much disk space for dbspaces as the Primary. The dbspace chunk types (cooked or raw) do not have to be identical
- Dbspace chunk path names must be identical, symbolic links can help here
- Not really a hardware requirement but any databases you want replicated must be logged. Unbuffered logging is preferred

Install and Configure Informix on the New Server

Follow the steps from Installing Innovator-C on Linux on a new server named blogsvr02. Create 0 byte files with the touch command to mirror the dbspace chunks on the primary server.

```
informix@blogsvr02> mkdir /home/informix/chunks
informix@blogsvr02> touch /home/informix/chunks/ROOTDBS.01
informix@blogsvr02> touch /home/informix/chunks/LLOGDBS01.01
informix@blogsvr02> touch /home/informix/chunks/DATADBS01.01
informix@blogsvr02> touch /home/informix/chunks/DATADBS01.02
infofmix@blogsvr02> chmod 660 /home/informix/chunks/*
```

Copy the /etc/profile.d/informix.sh file from the Primary to the Secondary and change INFORMIXSERVER

```
root@blogsvr02> scp blogsvr01:/etc/profile.d/informix.sh /etc/profile.d/informix.sh
root@blogsvr02> vi /etc/profile.d/informix.sh

export INFORMIXSERVER=blogsvr02
```

Copy the ONCONFIG file from the Primary to the Secondary and change DBSERVERNAME and add a DBSERVERALIASES to both ONCONFIGs that will be used exclusively for HDR.

```
informix@blogsvr01> vi $INFORMIXDIR/etc/$ONCONFIG

DBSERVERALIASES blogsvr01_hdr

informix@blogsvr02> scp blogsvr01:/opt/informix/etc/onconfig.blogsvr01 $INFORMIXDIR/etc/$ONCONFIG
informix@blogsvr02> vi $INFORMIXDIR/etc/$ONCONFIG

DBSERVERNAME blogsvr02
DBSERVERALIASES blogsvr02_hdr
```

Do we need a dedicated connection for HDR? No, but I feel doing so gives me two advantages

- I can put HDR traffic on a separate network if I want
- Both HDR servers must trust each other, I can use the more secure $INFORMIXDIR/etc/hosts.equiv to accomplish this if HDR runs on a dedicated port

If you would like to allow insert, update and deletes to take place on the Secondary server set the UPDATABLE_SECONDARY ONCONFIG parameter on both servers to a number between 1 and CPUVPs * 2 to configure the number of threads for transmitting updates from the Secondary to the Primary.

```
informix> vi $INFORMIXDIR/etc/$ONCONFIG

UPDATABLE_SECONDARY 2
```

Add a new port to /etc/services on both servers for HDR.

```
root> vi /etc/services

idshdr01         1528/tcp               # Informix HDR
```

Modify the sqlhosts file on both the Primary and the Secondary so they both contain connectivity information for both servers. Use the s=6 security option for the HDR ports to indicate that only Replication traffic is allowed on these ports giving us the ability to use $INFORMIXDIR/etc/hosts.equiv to establish trusts.

```
informix> vi $INFORMIXSQLHOSTS

# blogsvr01
blogsvr01              onsoctcp        blogsvr01               idstcp01
blogsvr01_hdr          onsoctcp        blogsvr01               idshdr01       s=6

# blogsvr02
blogsvr02              onsoctcp        blogsvr02               idstcp01
blogsvr02_hdr          onsoctcp        blogsvr02               idshdr01       s=6
```

Bounce the Primary server for ONCONFIG changes to take effect.

Create or Modify an Existing hosts.equiv Files

The hosts.equiv file will contain the hostname of each server that is allowed to make a trusted connection. You must also change the permissions of the file so only the informix user can write to it.

```
informix@blogsvr01> vi $INFORMIXDIR/etc/hosts.equiv

blogsvr02

informix@blogsvr01> chmod 640 $INFORMIXDIR/etc/hosts.equiv

informix@blogsvr02> vi $INFORMIXDIR/etc/hosts.equiv

blogsvr01

informix@blogsvr02> chmod 640 $INFORMIXDIR/etc/hosts.equiv
```

Note: Later when we start HDR if you see messages in your online.log (onstat -m output) that look like this:

```
12:12:16  listener-thread: err = -956: oserr = 0: errstr = informix@blogsvr02.prod.informix-
dba.com[blogsvr02]: Client host or user informix@blogsvr02.prod.informix-dba.com[blogsvr02] is not
trusted by the server.
```

then need you to put the full hostname, blogsvr02.prod.informix-dba.com, in hosts.equiv

Restore Secondary Server Using a Backup from the Primary

The first step in actually starting HDR is to perform a physical restore of the Primary to the Secondary. After this is complete we will start HDR and Informix will automatically sync the Secondary with the Primary by processing the logical log records that have been written since the Primary's backup was taken.

One of my favorite Informix features is ontape to STDIO, you can use this feature to simultaneously take a Level 0 backup of your Primary, ship the data over the network and pipe it directly into a physical restore on the Secondary. This is a lot easier than performing an Imported Restore. Like to see it? Here it goes.

```
informix@blogsvr01> ontape -s -L 0 -F -t STDIO | ssh informix@blogsvr02 ". /etc/profile.d/informix.sh;
ontape -p -t STDIO"
```

While this is running, you can use onstat -D on both servers to see the reading of pages on the Primary and the writing of pages on the Secondary in parallel. After the backup and restore completes the Secondary server will be in Fast Recovery mode.

```
informix@blogsvr02> onstat -m

IBM Informix Dynamic Server Version 12.10.UC7IE -- Fast Recovery -- Up 00:00:40 -- 1164976 Kbytes

Message Log File: /opt/informix-ids-12.10.UC7IE/tmp/online.log
13:38:11  Maximum server connections 0
13:38:11  Checkpoint Statistics - Avg. Txn Block Time 0.000, # Txns blocked 0, Plog used 0, Llog used 0

13:38:11  Checkpoint Completed:  duration was 0 seconds.
13:38:11  Tue Aug  3 - loguniq 10, logpos 0x1816018, timestamp: 0x4a722 Interval: 721

13:38:11  Maximum server connections 0
13:38:11  Checkpoint Statistics - Avg. Txn Block Time 0.000, # Txns blocked 0, Plog used 0, Llog used 0

13:38:11  Checkpoint Completed:  duration was 0 seconds.
13:38:11  Tue Aug  3 - loguniq 10, logpos 0x1816018, timestamp: 0x4a728 Interval: 722

13:38:11  Maximum server connections 0
13:38:11  Checkpoint Statistics - Avg. Txn Block Time 0.000, # Txns blocked 0, Plog used 0, Llog used 0

13:38:12  Physical Restore of rootdbs, llogdbs01, datadbs01 Completed.
13:38:12  Checkpoint Completed:  duration was 0 seconds.
13:38:12  Tue Aug  3 - loguniq 10, logpos 0x1816018, timestamp: 0x4a739 Interval: 722

13:38:12  Maximum server connections 0
```

and you are ready to start HDR.

Starting HDR

Start HDR on the Primary with the onmode -d primary command. In this command you will tell Informix that this is a Primary HDR server and the Secondary is blogsvr02.

```
informix@blogsvr01> onmode -d primary blogsvr02
```

Start HDR on the Secondary with the onmode -d secondary command. This will tell Informix that this is a Secondary HDR server and the Primary is blogsvr01.

```
informix@blogsvr02> onmode -d secondary blogsvr01
```

The two servers will connect and after the Secondary clears its logical logs and receives all of the logical log records from the Primary the HDR setup is complete.

```
informix@blogsvr02> onstat -m

IBM Informix Dynamic Server Version 12.10.UC7IE -- Updatable (Sec) -- Up 00:05:12 -- 1164976 Kbytes

Message Log File: /opt/informix-ids-12.10.UC7IE/tmp/online.log
13:42:09  Updates from secondary allowed
13:42:09  DR: Secondary server needs failure recovery

13:42:10  DR: Failure recovery from disk in progress ...
13:42:10  Logical Recovery Started.
13:42:10  10 recovery worker threads will be started.
13:42:10  Start Logical Recovery - Start Log 10, End Log ?
13:42:10  Starting Log Position - 10 0x1816018
13:42:10  Clearing the physical and logical logs has started
13:42:46  Cleared 3059 MB of the physical and logical logs in 36 seconds
13:42:48  Started processing open transactions on secondary during startup
13:42:48  Finished processing open transactions on secondary during startup.
13:42:48  DR: HDR secondary server operational
13:42:49  B-tree scanners disabled.
13:42:50  Checkpoint Completed:  duration was 0 seconds.
13:42:50  Tue Aug  3 - loguniq 10, logpos 0x181e018, timestamp: 0x4a7af Interval: 723

13:42:50  Maximum server connections 0
13:42:50  Checkpoint Statistics - Avg. Txn Block Time 0.000, # Txns blocked 0, Plog used 14, Llog used
0
```

You really don't have to do anything else from this point forward to administer HDR, just sit back and relax. You're data is safer now.

What do I do when the Secondary Server Fails?

If the Secondary fails and the logical log that was current at the time of the failure has not been reused (they're circular, remember) on the Primary then you can simply restart the Secondary and it will automatically resync.

```
informix@blogsvr02> oninit
informix@blogsvr02> tail -40 $INFORMIXDIR/tmp/online.log
14:46:39  DR: ENCRYPT_HDR is 0 (HDR encryption Disabled)
14:46:39  Event notification facility epoll enabled.
14:46:39  IBM Informix Dynamic Server Version 12.10.UC7IE Software Serial Number AAA#B000000
14:46:40  IBM Informix Dynamic Server Initialized -- Shared Memory Initialized.

14:46:40  Started 1 B-tree scanners.
14:46:40  B-tree scanner threshold set at 5000.
14:46:40  B-tree scanner range scan size set to -1.
14:46:40  B-tree scanner ALICE mode set to 6.
14:46:40  B-tree scanner index compression level set to med.
14:46:40  Physical Recovery Started at Page (1:5623).
14:46:40  Physical Recovery Complete: 0 Pages Examined, 0 Pages Restored.
14:46:40  DR: Trying to connect to primary server = blogsvr01_hdr
14:46:41  Dataskip is now OFF for all dbspaces
14:46:41  Restartable Restore has been ENABLED
14:46:41  Recovery Mode
```

```
14:46:45  DR: Secondary server connected
14:46:46  Updates from secondary allowed
14:46:46  Updates from secondary allowed
14:46:46  DR: Using default behavior of failure-recovering Secondary server

14:46:47  DR: Failure recovery from disk in progress ...
14:46:47  Logical Recovery Started.
14:46:47  10 recovery worker threads will be started.
14:46:47  Start Logical Recovery - Start Log 10, End Log ?
14:46:47  Starting Log Position - 10 0x182e018
14:46:48  Started processing open transactions on secondary during startup
14:46:48  Finished processing open transactions on secondary during startup.
14:46:48  DR: HDR secondary server operational
14:46:49  Logical Log 10 Complete, timestamp: 0x4a92d.
14:46:50  Logical Log 11 Complete, timestamp: 0x4a944.
14:46:51  Logical Log 12 Complete, timestamp: 0x4a975.
14:46:52  Logical Log 13 Complete, timestamp: 0x4a987.
14:46:54  B-tree scanners disabled.
14:46:55  Checkpoint Completed:  duration was 0 seconds.
14:46:55  Tue Aug  3 - loguniq 14, logpos 0x9018, timestamp: 0x4a9a4 Interval: 729

14:46:55  Maximum server connections 0
14:46:55  Checkpoint Statistics - Avg. Txn Block Time 0.000, # Txns blocked 0, Plog used 15, Llog used
0
```

If your Secondary has been down for a while and the logical logs have rolled over there are 2 ways to recover. The easy way and the hard way.

The easy way is to reinitialize HDR by restoring the Primary to the Secondary again and running onmode -d secondary blogsvr01_hdr on the Secondary.

The hard way is to restart the Secondary and when you see this message in the online.log

```
15:03:21  DR: Start failure recovery from tape ...
```

You can perform a Logical Restore to the Secondary using the logical log backups from the Primary. If you're backing up to a directory, copy the necessary logical log backups from the Primary to the Secondary, rename each backup to include the Secondary server name and use ontape -l -d to perform a Logical Restore.

```
informix@blogsvr02> scp blogsvr01:/home/informix/backup/llog/* .
blogsvr01_0_Log0000000008                                     100%   96KB
96.0KB/s   00:00
blogsvr01_0_Log0000000009                                     100%   96KB
96.0KB/s   00:00
blogsvr01_0_Log0000000010                                     100% 1440KB
1.4MB/s   00:00
blogsvr01_0_Log0000000011                                     100%   96KB
96.0KB/s   00:00
blogsvr01_0_Log0000000012                                     100%   96KB
96.0KB/s   00:00
blogsvr01_0_Log0000000013                                     100%   96KB
96.0KB/s   00:00
blogsvr01_0_Log0000000014                                     100%   96KB
96.0KB/s   00:00
blogsvr01_0_Log0000000015                                     100%   96KB
96.0KB/s   00:00
blogsvr01_0_Log0000000016                                     100%   96KB
96.0KB/s   00:00
blogsvr01_0_Log0000000017                                     100%   96KB
96.0KB/s   00:00
blogsvr01_0_Log0000000018                                     100%   96KB
96.0KB/s   00:00
blogsvr01_0_Log0000000019                                     100%   96KB
96.0KB/s   00:00
blogsvr01_0_Log0000000020                                     100%   96KB
96.0KB/s   00:00
blogsvr01_0_Log0000000021                                     100%   96KB
96.0KB/s   00:00
blogsvr01_0_Log0000000022                                     100%   96KB
96.0KB/s   00:00
```

```
blogsvr01_0_Log0000000023                                                    100%   96KB
96.0KB/s   00:00
blogsvr01_0_Log0000000024                                                    100%   96KB
96.0KB/s   00:00
blogsvr01_0_Log0000000025                                                    100%   96KB
96.0KB/s   00:00
blogsvr01_0_Log0000000026                                                    100%   96KB
96.0KB/s   00:00
blogsvr01_0_Log0000000027                                                    100%   96KB
96.0KB/s   00:00
blogsvr01_0_Log0000000028                                                    100%   96KB
96.0KB/s   00:00
blogsvr01_0_Log0000000029                                                    100%   96KB
96.0KB/s   00:00
blogsvr01_0_Log0000000030                                                    100%   96KB
96.0KB/s   00:00

informix@blogsvr02> script_i_made_to_rename_the_files.ksh
informix@blogsvr02> ls -l /home/informix/backup/llog
total 3648
-rw-rw---- 1 informix informix   98304 Aug  3 14:56 blogsvr02_0_Log0000000008
-rw-rw---- 1 informix informix   98304 Aug  3 14:56 blogsvr02_0_Log0000000009
-rw-rw---- 1 informix informix 1474560 Aug  3 14:56 blogsvr02_0_Log0000000010
-rw-rw---- 1 informix informix   98304 Aug  3 14:56 blogsvr02_0_Log0000000011
-rw-rw---- 1 informix informix   98304 Aug  3 14:56 blogsvr02_0_Log0000000012
-rw-rw---- 1 informix informix   98304 Aug  3 14:56 blogsvr02_0_Log0000000013
-rw-rw---- 1 informix informix   98304 Aug  3 14:56 blogsvr02_0_Log0000000014
-rw-rw---- 1 informix informix   98304 Aug  3 14:56 blogsvr02_0_Log0000000015
-rw-rw---- 1 informix informix   98304 Aug  3 14:56 blogsvr02_0_Log0000000016
-rw-rw---- 1 informix informix   98304 Aug  3 14:56 blogsvr02_0_Log0000000017
-rw-rw---- 1 informix informix   98304 Aug  3 14:56 blogsvr02_0_Log0000000018
-rw-rw---- 1 informix informix   98304 Aug  3 14:56 blogsvr02_0_Log0000000019
-rw-rw---- 1 informix informix   98304 Aug  3 14:56 blogsvr02_0_Log0000000020
-rw-rw---- 1 informix informix   98304 Aug  3 14:56 blogsvr02_0_Log0000000021
-rw-rw---- 1 informix informix   98304 Aug  3 14:56 blogsvr02_0_Log0000000022
-rw-rw---- 1 informix informix   98304 Aug  3 14:56 blogsvr02_0_Log0000000023
-rw-rw---- 1 informix informix   98304 Aug  3 14:56 blogsvr02_0_Log0000000024
-rw-rw---- 1 informix informix   98304 Aug  3 14:56 blogsvr02_0_Log0000000025
-rw-rw---- 1 informix informix   98304 Aug  3 14:56 blogsvr02_0_Log0000000026
-rw-rw---- 1 informix informix   98304 Aug  3 14:56 blogsvr02_0_Log0000000027
-rw-rw---- 1 informix informix   98304 Aug  3 14:56 blogsvr02_0_Log0000000028
-rw-rw---- 1 informix informix   98304 Aug  3 14:56 blogsvr02_0_Log0000000029
-rw-rw---- 1 informix informix   98304 Aug  3 14:56 blogsvr02_0_Log0000000030

informix@blogsvr02> ontape -l -d
Roll forward should start with log number 14
Restore is using file /home/informix/backup/llog/blogsvr02_0_Log0000000014 ...
Using the backup and restore filter /bin/gunzip.
Rollforward log file /home/informix/backup/llog/blogsvr02_0_Log0000000014 ...
Using the backup and restore filter /bin/gunzip.
Rollforward log file /home/informix/backup/llog/blogsvr02_0_Log0000000015 ...
Using the backup and restore filter /bin/gunzip.
Rollforward log file /home/informix/backup/llog/blogsvr02_0_Log0000000016 ...
Using the backup and restore filter /bin/gunzip.
Rollforward log file /home/informix/backup/llog/blogsvr02_0_Log0000000017 ...
Using the backup and restore filter /bin/gunzip.
Rollforward log file /home/informix/backup/llog/blogsvr02_0_Log0000000018 ...
Using the backup and restore filter /bin/gunzip.
Rollforward log file /home/informix/backup/llog/blogsvr02_0_Log0000000019 ...
Using the backup and restore filter /bin/gunzip.
Rollforward log file /home/informix/backup/llog/blogsvr02_0_Log0000000020 ...
Using the backup and restore filter /bin/gunzip.
Rollforward log file /home/informix/backup/llog/blogsvr02_0_Log0000000021 ...
Using the backup and restore filter /bin/gunzip.
Rollforward log file /home/informix/backup/llog/blogsvr02_0_Log0000000022 ...
Using the backup and restore filter /bin/gunzip.
Rollforward log file /home/informix/backup/llog/blogsvr02_0_Log0000000023 ...
Using the backup and restore filter /bin/gunzip.
Rollforward log file /home/informix/backup/llog/blogsvr02_0_Log0000000024 ...
Using the backup and restore filter /bin/gunzip.
Rollforward log file /home/informix/backup/llog/blogsvr02_0_Log0000000025 ...
Using the backup and restore filter /bin/gunzip.
Rollforward log file /home/informix/backup/llog/blogsvr02_0_Log0000000026 ...
```

```
Using the backup and restore filter /bin/gunzip.
Rollforward log file /home/informix/backup/llog/blogsvr02_0_Log0000000027 ...
Using the backup and restore filter /bin/gunzip.
Rollforward log file /home/informix/backup/llog/blogsvr02_0_Log0000000028 ...
Using the backup and restore filter /bin/gunzip.
Rollforward log file /home/informix/backup/llog/blogsvr02_0_Log0000000029 ...
Using the backup and restore filter /bin/gunzip.
Rollforward log file /home/informix/backup/llog/blogsvr02_0_Log0000000030 ...

Program over.

informix@blogsvr02> tail -46 $INFORMIXDIR/tmp/online.log
15:03:21  DR: Start failure recovery from tape ...
15:03:28  Logical Recovery Started.
15:03:28  10 recovery worker threads will be started.
15:03:28  Start Logical Recovery - Start Log 14, End Log ?
15:03:28  Starting Log Position - 14 0x9018
15:03:29  Started processing open transactions on secondary during startup
15:03:29  Finished processing open transactions on secondary during startup.
15:03:29  DR: HDR secondary server operational
15:03:29  Logical Log 14 Complete, timestamp: 0x4a9e2.
15:03:29  Logical Log 15 Complete, timestamp: 0x4a9f9.
15:03:29  Logical Log 16 Complete, timestamp: 0x4aa0b.
15:03:29  Logical Log 17 Complete, timestamp: 0x4aa0b.
15:03:29  Logical Log 18 Complete, timestamp: 0x4aa33.
15:03:29  Logical Log 19 Complete, timestamp: 0x4aa45.
15:03:29  Logical Log 20 Complete, timestamp: 0x4aa45.
15:03:29  Logical Log 21 Complete, timestamp: 0x4aa6a.
15:03:29  Logical Log 22 Complete, timestamp: 0x4aa6a.
15:03:29  Logical Log 23 Complete, timestamp: 0x4aa94.
15:03:29  Logical Log 24 Complete, timestamp: 0x4aaa6.
15:03:29  Logical Log 25 Complete, timestamp: 0x4aaa6.
15:03:29  Logical Log 26 Complete, timestamp: 0x4aace.
15:03:29  Logical Log 27 Complete, timestamp: 0x4aace.
15:03:29  Logical Log 28 Complete, timestamp: 0x4aaf2.
15:03:29  Checkpoint Completed:  duration was 0 seconds.
15:03:29  Tue Aug  3 - loguniq 29, logpos 0x18, timestamp: 0x4aafc Interval: 730

15:03:29  Maximum server connections 0
15:03:29  Checkpoint Statistics - Avg. Txn Block Time 0.000, # Txns blocked 0, Plog used 16, Llog used
0

15:03:30  Logical Log 29 Complete, timestamp: 0x4ab1f.
15:03:33  DR: Failure recovery from disk in progress ...
15:03:33  Logical Log 30 Complete, timestamp: 0x4ae61.
15:03:33  Checkpoint Completed:  duration was 0 seconds.
15:03:33  Tue Aug  3 - loguniq 31, logpos 0x15018, timestamp: 0x4aea4 Interval: 731

15:03:33  Maximum server connections 0
15:03:33  Checkpoint Statistics - Avg. Txn Block Time 0.000, # Txns blocked 0, Plog used 13, Llog used
0

15:03:33  Checkpoint Completed:  duration was 0 seconds.
15:03:33  Tue Aug  3 - loguniq 31, logpos 0x17018, timestamp: 0x4aeaa Interval: 732

15:03:33  Maximum server connections 0
15:03:33  Checkpoint Statistics - Avg. Txn Block Time 0.000, # Txns blocked 0, Plog used 0, Llog used 0

15:03:35  B-tree scanners disabled.
15:03:36  Checkpoint Completed:  duration was 0 seconds.
15:03:36  Tue Aug  3 - loguniq 31, logpos 0x20018, timestamp: 0x4aed2 Interval: 733
```

What do I do when the Primary Fails?

When your Primary fails you can quickly make the Secondary server a Standalone (i.e. no HDR) server. Even if you have configured an Updatable Secondary you will need to do this since the writes on a Secondary are sent to the Primary under the covers.

Make the Secondary a Standalone server with the onmode -d standard command

```
informix@blogsvr02> onmode -d standard
```

```
informix@blogsvr02> onstat -m

IBM Informix Dynamic Server Version 12.10.UC7IE -- On-Line -- Up 00:30:32 -- 1164976 Kbytes

Message Log File: /opt/informix-ids-12.10.UC7IE/tmp/online.log
15:38:26  Logical Recovery Complete.
15:38:27  Quiescent Mode
15:38:27  Checkpoint Completed:  duration was 0 seconds.
15:38:27  Tue Aug  3 - loguniq 31, logpos 0x4c018, timestamp: 0x4b0a6 Interval: 740

15:38:27  Maximum server connections 0
15:38:27  Checkpoint Statistics - Avg. Txn Block Time 0.000, # Txns blocked 0, Plog used 0, Llog used 1

15:38:27  Started 1 B-tree scanners.
15:38:27  B-tree scanner threshold set at 5000.
15:38:27  B-tree scanner range scan size set to -1.
15:38:27  B-tree scanner ALICE mode set to 6.
15:38:27  B-tree scanner index compression level set to med.
15:38:27  DR: Reservation of the last logical log for log backup turned on
15:38:27  SCHAPI: Started dbScheduler thread.
15:38:27  DR: new type = standard
15:38:27  Booting Language  from module <>
15:38:27  Loading Module
15:38:27  SCHAPI: Started 2 dbWorker threads.
15:38:28  On-Line Mode
```

When the old Primary is fixed and ready to be brought back online you have options.

Option 1 is to reinitialize HDR just like we did when setting up HDR for the first time. Except now blogsvr02 will be the Primary and blogsvr01 will be the Secondary. I like this option because it doesn't require any downtime.

Option 2 is to make blogsvr01 the Primary again (easier to do if the logs have not rolled over on blogsvr02.) This requires some downtime and assumes that the disks on blogsvr01 were not the reason it went down and all of the data is still intact.

Switch blogsvr02 to Quiescent Mode

```
informix@blogsvr02> onmode -s
```

Change the HDR status of blogsvr02 to Secondary

```
informix@blogsvr02> onmode -d secondary blogsvr01_hdr
```

Start Informix on the Primary

```
informix@blogsvr01> oninit
```

If the logical logs have rolled over on the Secondary (while it was Standalone) you will need to do what we did before. Move the logical log backups that you need from blogsvr02 to blogsvr01, change their names and run ontape -l -d

If everything works as advertised the Secondary will ship over the logs the Primary needs, they will be applied to the Primary and HDR will be restored.

Pretty cool stuff that has saved my butt more than a couple of times.

Informix Tuning Basics

A follow up to the Informix Innovator-C Quick Start Guide to help the new Informix DBA perform basic performance tuning. As with the Quick Start Guide the title references Innovator-C because I'm assuming this is what the new Informix users are using, but most content is relevant to almost all recent versions of Informix.

In this Guide I will:

- Assume only 2 KB dbspaces
- Initially configure Buffers and LRU Queues

- Monitor and tune Buffers
- Monitor and tune LRU Queues
- Monitor and tune writes
- Initially configure CPUVPs
- Monitor and Tune CPUVPs
- Initially configure AIOVPs
- Monitor and Tune AIOVPs

Increase Buffers to Reduce Disk I/O

Original Blog Post: Increasing Buffers and Reducing I/O with Informix

End Result:

- Buffers increased from default value of 50000
- LRUs increased from default value of 8

1. Determine how much memory you want Informix to allocate

If you are running Innovator-C this maximum is 2 GB, other Informix editions may have different restrictions.

You will also be restricted by the actual amount of memory available on your system. Some memory will be needed for the OS and some will be needed by the other applications running on your system. Identify the total memory footprint you want Informix to have and set the ONCONFIG parameter SHMTOTAL to this value in KB.

```
informix> vi $INFORMIXDIR/etc/$ONCONFIG

SHMTOTAL 2097152
```

2. Determine how much of this memory you want Informix to allocate to Buffers.

Informix will cache the data that is on disk in Buffers reducing I/O and improving your performance. Don't be stingy. A decent starting point is 50%, you can adjust this later based on monitoring actual server activity.

Buffer Memory = SHMTOTAL * Buffer Percent of Memory
Buffer Memory = 2097152 KB * 0.5 = 1048576 KB

3. Calculate the number of 2 KB Buffers

Number of Buffers = 1048576 KB / 2 KB = 524288 Buffers

4. Pick a starting point for number of LRU Queues. I like to use 8322 Buffers per LRU, which gives 63 for 524288 Buffers.

Number of LRU Queues = Number of Buffers / 8322
Number of LRU Queues = 524288 / 8322 = 63.0002 = 63 LRU Queues

5. Change the ONCONFIG BUFFERS parameter and bounce the engine to make change take effect

```
informix> vi $INFORMIXDIR/etc/$ONCONFIG

BUFFERPOOL size=2k,buffers=524288,lrus=63,lru_min_dirty=50,lru_max_dirty=60

informix> onmode -ky
informix> oninit -v
```

<="" a="" style="color: rgb(143, 102, 102); text-decoration: none;">**Monitor and Tune Buffers**

Original Blog Post: Increasing Buffers and Reducing I/O with Informix

End Result:

- Determine if too many or not enough buffers are configured

1. Do not attempt to tune buffers until your engine has been running under normal load for some time.

At engine startup the buffer pool is empty and pages are brought into memory over time as SQL requests are processed. Attempting to tune the buffer pool too soon after engine startup will lead to inaccurate results.

2. Determine if too many buffers are configured by using onstat -P | tail get an summary of what is in the buffer pool

```
informix> onstat -P
7340033  4        0        0        4        0
7340034  3967     0        3967     0        0

Totals:  1048576  362651   682521   3404     7958

Percentages:
Data  55.09
Btree 24.59
Other 20.32
```

What is important here is the last 3 lines of the onstat -P output. This is telling me that 55.09% of my buffers contain data pages, 24.59% of my buffers contain index pages and 20.32% of my buffers contain other pages. A large portion of the other pages will be unused buffers.

From this information I see that 20% of my buffer pool is going unused and I can safely reduce the number of buffers by 20%.

3. Determine if not enough buffers are configured by executing onstat -p to get the Read Cache Hit %

The Read Cache Hit % is the percentage of times a page on disk was needed and was already in the buffer pool and is the first %cached column of the onstat -p output.

```
IBM Informix Dynamic Server Version 12.10.UC7IE   -- On-Line (Prim) -- Up 4 days 21:36:22 -- 1510960 Kbytes

Profile
dskreads   pagreads   bufreads   %cached dskwrits   pagwrits   bufwrits   %cached
363387     10087525   2903812263 99.99   1122317    1294778    14678622   92.35

isamtot    open       start      read       write      rewrite    delete     commit     rollbk
2473713077 116899849  241695271  1638363472 1447480    443270     48433      721760     218019

gp_read    gp_write   gp_rewrt   gp_del     gp_alloc   gp_free    gp_curs
0          0          0          0          0          0          0

ovlock     ovuserthread ovbuff     usercpu  syscpu   numckpts   flushes
0          0            0          11955.10 906.98   30         16

bufwaits   lokwaits   lockreqs   deadlks    dltouts    ckpwaits   compress   seqscans
7280       300        2020820205 0          0          10         449429     208087
```

```
ixda-RA    idx-RA     da-RA      RA-pgsused lchwaits
62769      5239       1667       69665      846929
```

In the above output my Read Cache Hit % is 99.99%, hard to improve on that. Anything above 95% is great, anything above 90% is good and anything below 90% is a good indication of needing more buffers.

If adding more buffers does not improve your Read Cache then it is possible you have some poorly optimized SQLs that could be using sequential scans which can flood the buffers with a lot of pages that are not reused. If this is the case look at creating necessary indexes to avoid sequential scans or look into using light scans to avoid using the buffer pool for these sequential scans.

You can also utilize the Read Cache Hit % to determine if too many buffers are configured. If you need to reduce the amount of shared memory used by Informix you can most likely reduce the number of buffers if your hit rate is very high. This will result in poorer performance but as long as you keep it above 95% you should be OK.

Monitor LRU Queues

Original Blog Post: Increasing Buffers and Reducing I/O with Informix

End Result:

- Determine if enough LRU Queues are configured

1. Calculate Bufwaits Ratio (courtesy of Art Kagel) using bufwaits, pagreads and bufwrits from the onstat -p output

```
IBM Informix Dynamic Server Version 12.10.UC7IE   -- On-Line (Prim) -- Up 4 days 21:36:22 -- 1510960 Kbytes

Profile
dskreads   pagreads   bufreads   %cached dskwrits   pagwrits   bufwrits   %cached
363387     10087525   2903812263 99.99   1122317    1294778    14678622   92.35

isamtot    open       start      read       write      rewrite    delete     commit     rollbk
2473713077 116899849  241695271  1638363472 1447480    443270     48433      721760     218019

gp_read    gp_write   gp_rewrt   gp_del     gp_alloc   gp_free    gp_curs
0          0          0          0          0          0          0

ovlock     ovuserthread ovbuff     usercpu  syscpu   numckpts   flushes
0          0            0          11955.10 906.98   30         16

bufwaits   lokwaits   lockreqs   deadlks    dltouts    ckpwaits   compress   seqscans
7280       300        2020820205 0          0          10         449429     208087

ixda-RA    idx-RA     da-RA      RA-pgsused lchwaits
62769      5239       1667       69665      846929
```

bufwaits = 7280
pagreads = 10087525
bufwrits = 14678622

Bufwaits Ratio (BR) = (bufwaits / (pagreads + bufwrits)) * 100
Bufwaits Ratio (BR) = (7280 / (10087525 + 14678622)) * 100 = 0.029%

2. Increase LRU Queues by 10% until BR is less than 7% or the maximum number of LRUs are configured

Maximum LRU Queues for 32 bit engines is 128
Maximum LRU Queues for 64 bit engines is 512

Monitor and Tune Disk Writes

Original Blog Post: Increasing Buffers and Reducing I/O with Informix

End Result:

- Maximize Checkpoint writes
- Minimize Foreground writes

1. Use onstat -F output to determine if any Foreground writes are taking place

```
informix> onstat -F | head

IBM Informix Dynamic Server Version 12.10.UC7IE   -- On-Line (Prim) -- Up 4 days 22:09:56 -- 1510960 Kbytes


Fg Writes     LRU Writes    Chunk Writes
1093          98032         208755

address          flusher  state    data     # LRU    Chunk    Wakeups  Idle Tim
db957880         0        I        0        0        366      425136   425380.489
db9580d8         1        I        0        0        362      424787   424905.846
```

In the above output there have been 1093 Foreground (FG) writes, 98032 LRU writes and 208755 Chunk writes (aka Checkpoint writes).

2. If Foreground writes are occurring then you need to increase LRU writes to keep some free buffers (aka buffers that do not need to be sync'd to disk at some point) in the LRU queues.

Reduce the LRU Max Dirty and LRU Min Dirty by modifying the BUFFERS ONCONFIG parameter until FG writes no longer occur.

3. If Foregound writes are not occurring then you may be interested in increasing LRU Max Dirty and LRU Min Dirty to reduce the number of LRU writes and increase the number of Chunk writes (which are more efficient) at checkpoint time.

One word of caution. Even though Checkpoint writes are more efficient than LRU writes you must understand what is happening when you maximize Checkpoint writes. When a checkpoint occurs there is a lot of I/O as Informix writes out all of the modified pages to disk. This I/O results in longer run times for SQLs executing during the checkpoint (hey, at least they aren't blocking anymore!) Moving the maximum number of writes to checkpoint time will increase the total number of transactions you can perform because the engine will be more efficient, but the checkpoint times will be longer causing more SQLs to run during the slower period.

You might not run into the problem at all, it is just something to be aware of when configuring writes.

Initially configure CPUVPs

Original Blog Post: Configuring Informix Virtual Processors

End Result:

- CPU Virtual Processors initially configured based on your hardware
- Assumes all CPU resources can be consumed by Informix. If this is not the case, reduce CPUVPs accordingly.

1. Identify the number of physical CPU cores. Do not count a Hyperthreaded CPU as having 2 cores.

2. Identify the maximum number of CPUVPs you can configure in your Informix Edition

- Innovator-C: 4 CPUVPs
- Choice: 8 CPUVPs
- Growth: 16 CPUVPs
- Ultimate: No limit

3. Configure MULTIPROCESSOR ONCONFIG parameter

1 core systems: set MULTIPROCESSOR to 0.
Everything else: set MULTIPROCESSOR to 1.

4. Configure SINGLE_CPU_VP ONCONFIG parameter

1 and 2 core systems: set SINGLE_CPU_VP to 1.
Everything else: set SINGLE_CPU_VP to 0.

5. Configure number of CPUVPs using VPCLASS ONCONFIG parameter

1 and 2 core systems: Number of CPUVPs = 1
Everything else: Number of CPUVPs = Number of cores - 1. Do not exceed the limits of your Informix Edition.

For an 8 core system running Innovator-C, configure 4 CPUVPs

```
VPCLASS cpu,num=4,noage
```

6. Shutdown and restart Informix for changes to take effect.

Monitor and Tune CPUVPs

Original Log Post: Configuring Informix Virtual Processors

End Result:

- Number of CPUVPs tuned to match Informix workload and CPU resources

1. After Informix has been running under normal load for a few days use onstat -g glo to monitor the efficiency of each CPUVP.

WARNING: onstat -g glo produces incorrect efficiency in versions prior to 12.10.xC6 if onstat -z is used to clear statistics!

Efficiency is the percentage of time a VP's threads were actually running on a CPU vs the time they where scheduled to run by Informix. A high efficiency means Informix is not waiting for the CPU to become available to do work. The efficiency is contained in the 'Individual virtual processors' output on onstat -g glo.

```
informix> onstat -g glo

IBM Informix Dynamic Server Version 12.10.UC7IE -- On-Line -- Up 2 days 01:00:48 -- 1410736 Kbytes

...

Individual virtual processors:
 vp    pid     class     usercpu   syscpu    total     Thread    Eff
 1     7461    cpu       8012.55   110.13    8122.68   8291.89   97%
 2     7464    adm       0.06      1.21      1.27      0.00      0%
 3     7486    lio       5.57      22.67     28.24     283.83    9%
 4     7495    pio       0.02      0.31      0.33      8.79      3%
 5     7497    aio       5.78      27.54     33.32     114.46    29%
```

```
6    7499     msc       0.00     0.00     0.00     0.02      0%
7    7501     fifo      0.00     0.00     0.00     0.00      0%
8    7503     aio       0.95     2.12     3.07     26.92    11%
9    7526     soc       0.18     0.36     0.54     NA        NA
10   7538     aio       0.05     0.52     0.57     31.09     1%
11   7539     aio       0.05     0.59     0.64     27.98     2%
12   7540     aio       0.04     0.14     0.18     13.48     1%
13   7541     aio       0.03     0.49     0.52     16.06     3%
14   29965    aio       0.03     0.10     0.13     15.82     0%
15   13400    aio       0.03     0.34     0.37     20.00     1%
              tot       8025.34  166.52   8191.86
```

2. If your efficiency is high you might benefit from additional CPUVPs.

To determine if additional CPUVPs will improve performance you must monitor the CPUVP ready queue. This is where threads are placed when they are ready to be run by a CPUVP. Threads consistently in the ready queue indicate there are not enough CPUVPs to handle the current workload.

You monitor ready queue with the onstat -g rea command. Use onstat -r 1 -g rea to display the ready queue every second. A system that might benefit from more CPUVPs will look like this:

```
informix> onstat -r 1 -g rea

IBM Informix Dynamic Server Version 12.10.UC7IE -- On-Line -- Up 2 days 01:12:27 -- 1418928 Kbytes

Ready threads:
 tid     tcb      rstcb    prty status            vp-class      name
 292     e41290d0 e12f9f30 1    ready             1cpu          sqlexec


IBM Informix Dynamic Server Version 12.10.UC7IE -- On-Line -- Up 2 days 01:12:28 -- 1418928 Kbytes

Ready threads:
 tid     tcb      rstcb    prty status            vp-class      name
 290     e4127afe e12f2b40 1    ready             1cpu          sqlexec
 292     e41290d0 e12f9f30 1    ready             1cpu          sqlexec

IBM Informix Dynamic Server Version 12.10.UC7IE -- On-Line -- Up 2 days 01:12:29 -- 1418928 Kbytes

Ready threads:
 tid     tcb      rstcb    prty status            vp-class      name
 290     e4127afe e12f2b40 1    ready             1cpu          sqlexec

IBM Informix Dynamic Server Version 12.10.UC7IE -- On-Line -- Up 2 days 01:12:30 -- 1418928 Kbytes

Ready threads:
 tid     tcb      rstcb    prty status            vp-class      name
 294     e412bce3 e123da45 1    ready             1cpu          sqlexec
 290     e4127afe e12f2b40 1    ready             1cpu          sqlexec
```

3. If your efficiency is low you may have too many CPUVPs configured or your CPUs can not handle the load

Reduce the number of CPUVPs, add additional CPUs or move applications that are also running on this server and taking CPU resources from Informix to a different server.

Initially configure AIOVPs

Original Blog Post: Configuring Informix Virtual Processors

End Result:

- AIO Virtual Processors initially configured based on your hardware

1. If Kernel Asynchronous IO is enabled

Number of AIOVPs = Number of physical disks containing cooked chunks + 2

2. If Kernel Asynchronous IO is disabled

Number of AIOVPs = Number of physical disks containing chunks + 2

3. Modify the VPCLASS ONCONFIG parameter for AIOVPs and bounce Informix

VPCLASS aio,num=5,noage

Monitor and Tune AIOVPs

Original Blog Post: Configuring Informix Virtual Processors

End Result:

- AIO Virtual Processors dynamically tuned to meet your I/O demand
- Determine if AIOVPs can handle I/O load

1. Enable automatic AIOVP tuning which will dynamically add AIOVPs if they are needed

```
informix> vi $INFORMIXDIR/etc/$ONCONFIG

AUTO_AIOVPS 1
```

2. Use onstat -g iov to determine if I/O requests are backing up

onstat -g iov will show us the number of I/Os performed per AIOVP wakeup. We want to see one or more of the most active AIOVPs with an io/wup of 1.0 or less which tells us on the average number of I/O requests are in the queue each time an AIOVP was awakened to do some I/O work. If there is consistently only 1 I/O request in the queue then our AIOVPs are handling the I/O load.

```
informix> onstat -g iov

IBM Informix Dynamic Server Version 12.10.UC7IE   -- On-Line (Prim) -- Up 5 days 11:11:11 -- 1510960 Kbytes

AIO I/O vps:
class/vp/id s  io/s totalops  dskread dskwrite  dskcopy  wakeups  io/wup  errors tempops
 fifo  9  0 i   0.0        0        0        0        0        0   0.0       0        0
  msc  8  0 i   0.2   343147        0        0        0   343081   1.0       0   343170
  aio  7  0 i  62.0  4761188  3052788   102515        0  4680405   1.0       0    23338
  aio 10  1 i   7.9   604355   487107    38564        0   513321   1.2       0     1036
  aio 11  2 i   3.0   227818   166585    30099        0   180650   1.3       0      222
  aio 12  3 i   1.7   130962    98601    26027        0    88725   1.5       0        7
  aio 13  4 i   1.4   107353    81510    24218        0    69842   1.5       0        5
  aio 14  5 i   1.2    95549    72486    22105        0    61630   1.6       0        5
  aio 15  6 i   1.1    87313    65094    21449        0    57238   1.5       0        3
  aio 16  7 i   1.1    82456    61836    19979        0    54071   1.5       0        3
  aio 17  8 i   1.0    77195    57215    19367        0    49859   1.5       0        4
  aio 18  9 i   0.9    70738    51898    18305        0    45489   1.6       0        5
  pio  6  0 i   0.1     4326        0     4326        0     4326   1.0       0    37156
  lio  5  0 i  31.3  2405192        0  2405192        0  2396927   1.0       0 19652116
```